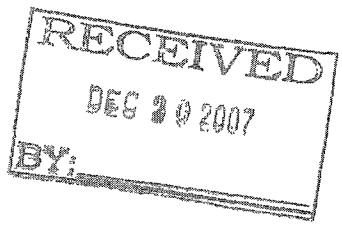
Exhibit K-1

103,1002.13



M/S Inter Partes Reexam
Commissioner for Patents
P.O. Box 1450,
Alexandria, VA 22313-1450

US Patent: 6,892,211 By: RLY:rac
Issued: May 10, 2005
DLAP File No.: 347155-29

Title: Copy On Write File System Consistency And Block Usage

Mail Date: December 14, 2007 Due Date: N/A

The following have been received in the U.S. Patent and Trademark Office on the date stamped hereon:

1. Transmittal Form;
2. Request for Inter Parte Reexamination Transmittal Form;
3. Attachment to Request for Inter-Partes Re-Examination of U.S. Patent No. 6,892,211; and
4. Information Disclosure Statement & PTO-1449; w/13 references;
5. Certificate of Mailing By Express Mail No.: EV 978 428 088 US; and
6. Return Post Card.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

EXPRESS MAIL NUMBER: EV 978 428 088 US DATE OF DEPOSIT: December 14, 2007

I hereby certify that this paper is being deposited with the United States Postal Service "EXPRESS MAIL Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Commissioner for Patents; P.O. Box 1450, Alexandria, VA 22313-1450.

Rosa A. Caviedes

Rosa A. Caviedes

* * *

CERTIFICATE OF MAILING BY EXPRESS MAIL

COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-14500

Sir:

Transmitted herewith for filing is the following:

1. Transmittal Form;
2. Request for Inter Partes Reexamination Transmittal Form (+1 copy);
3. Attachment to Request Inter-Partes Re-Examination of U.S. Patent No. 6,892,211;
4. Information Disclosure Statement & PTO-1449 w/13 references;
5. Certificate of Mailing By Express Mail No.: EV 978 428 088 US; and
6. Return Post Card.

EV978428088US

PTO/SB/21 (10-07)
Approved for use through 10/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission	

U.S. Patent No.	6,892,211
Filing Date	May 10, 2005
First Named Inventor	Hitz et al.
Art Unit	N/A
Examiner Name	N/A
Attorney Docket Number	347155-29

ENCLOSURES *(Check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment/Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☒ Information Disclosure Statement & PTO-1449 with 13 references
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
  - ☐ Reply to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☒ Request for Inter Parte Reexamination Transmittal Form
- ☐ CD, Number of CD(s) _____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) (please Identify below):
  1. Attachment to Request for Inter-Partes Re-Examination of U.S. Patent No. 6,892,211; and
  2 Return Post Card.

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	DLA Piper US LLP		
Signature	Edward Wolfe (Reg. No. 37,468) for RLY		
Printed name	Ronald L. Yin		
Date	December 14, 2007	Reg. No.	27,607

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage via "EXPRESS MAIL No. EV 978 428 088 US in an envelope addressed to: Commissioner for Patents, MS BOX REEXAM, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	Edward Wolfe (Reg. No. 37,468) for RLY		
Typed or printed name	Ronald L. Yin	Date	December 14, 2007

PTO/SB/58 (09-07)
Approved for use through 08/31/2010. OMB 0651-0033
U.S Patent and Trademark Office, U S DEPARTMENT OF COMMERCE

(Also referred to as FORM PTO-1465)

REQUEST FOR *INTER PARTES* REEXAMINATION TRANSMITTAL FORM

Address to:
Mail Stop *Inter Partes* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: 347155-29

Date: December 14, 2007

1 [X] This is a request for *inter partes* reexamination pursuant to 37 CFR 1.913 of patent number 6,892,211 issued May 10, 2005. The request is made by a third party requester, identified herein below.

2. [X] a The name and address of the person requesting reexamination is

Ronald L. Yin
DLA Piper US LLP
2000 University Avenue, East Palo Alto, CA 94303

b. The real party in interest (37 CFR 1.915(b)(8)) is: Sun Microsystems, Inc.

3 [ ] a. A check in the amount of $ _____ is enclosed to cover the reexamination fee, 37 CFR 1 20(c)(2),

[X] b The Director is hereby authorized to charge the fee as set forth in 37 CFR 1 20(c)(2) to Deposit Account No. 07-1896 (submit duplicative copy for fee processing), or

[ ] c Payment by credit card. Form PTO-2038 is attached.

4. [X] Any refund should be made by [ ] check or [X] credit to Deposit Account No 07-1896 37 CFR 1.26(c) If payment is made by credit card, refund must be made to credit card account

5. [X] A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.915(b)(5)

6 [ ] CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table
[ ] Landscape Table on CD

7 [ ] Nucleotide and/or Amino Acid Sequence Submission
If applicable, items a – c are required

a. [ ] Computer Readable Form (CRF)
b. Specification Sequence Listing on
i [ ] CD-ROM (2 copies) or CD-R (2 copies), **or**
ii [ ] paper
c [ ] Statements verifying identity of above copies

8. [X] A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.

9. [X] Reexamination of claim(s) 1-24 is requested

10 [X] A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.

11. [X] An English language translation of all necessary and pertinent non-English language patents and/or printed publications is included.

PTO/SB/58 (09-07)
Approved for use through 08/31/2010. OMB 0651-0033
U S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

12. [X] The attached detailed request includes at least the following items:

a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.915(b)(3)
b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR 1.915(b)(1) and (3)

13. [X] It is certified that the estoppel provisions of 37 CFR 1.907 do not prohibit this reexamination. 37 CFR 1.915(b)(7)

14 [X] a. It is certified that a copy of this request has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).
The name and address of the party served and the date of service are.

Steven A. Swernofsky

Swernofsky Law Group PC

P.O. Box 390013, Mountain View, CA 94039-0013

Date of Service: December 14, 2007; or

[ ] b A duplicate copy is enclosed since service on patent owner was not possible.

15. Correspondence Address Direct all communications about the application to:

[X] The address associated with Customer Number. 26379

OR

[ ] Firm or Individual Name

Address

City	State	Zip

Country

Telephone	Email

16. [X] The patent is currently the subject of the following concurrent proceeding(s):

[ ] a Copending reissue Application No _____
[ ] b. Copending reexamination Control No _____
[ ] c. Copending Interference No. _____.
[X] d. Copending litigation styled

Network Appliance, Inc. v. Sun Microsystems, Inc.

C-07-06053 EDL USDC ND CA (SF Div)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Ronald L. Yin — December 14, 2007

Authorized Signature For Third Party Requester — Date

Ronald L. Yin — 27,607

Typed/Printed Name — Registration Number, if applicable

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Hitz et al.	
U.S. Patent No.	6,892,211	Issued: May 10, 2005
Filed:	April 12, 2004	
Docket No.:	347155-29	
Title:	COPY ON WRITE FILE SYSTEM CONSISTENCY AND BLOCK USAGE	

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage Via Express Mail No. EV 978 428 088 US US in an envelope addressed to: Commissioner of Patents, MS Inter Partes REEXAM, P.O. Box 1450, Alexandria, VA 22313-1450, on:
December 14, 2007
Ronald L. Yin

* * *

ATTACHMENT TO REQUEST FOR INTER-PARTES RE-EXAMINATION OF U.S. PATENT NO. 6,892,211

Mail Stop *Inter Partes* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA. 22313-1450

Sir:

Pursuant to 35 U.S.C. §§ 311-318 and 37 CFR § 1.903-1.997, this is a request for inter-partes reexamination of United States Patent No. 6,892,211 which issued on May 10, 2005 to Hitz et al. (the "'211 Patent").

I. CLAIMS FOR WHICH REEXAMINATION IS REQUESTED

Reexamination is requested of Claims 1-24 of the '211 Patent in view of the prior art listed on the Citation of Prior Art under 37 CFR § 1.501 and 35 U.S.C. § 301 which is submitted with the Request for Reexamination.

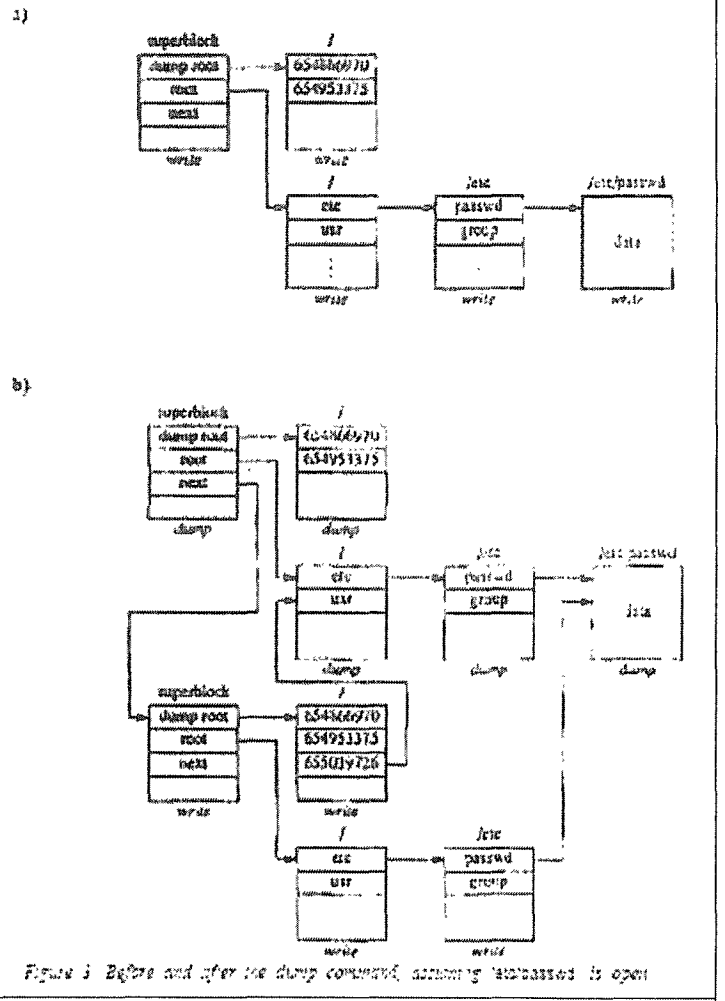
First Basis of Invalidity

The reference applicable to the first basis of invalidity is:

1. Quinlan, *A Cached WORM File System*, Software – Practice And Experience, Vol. 21(12), 1289-1299, December 1991 ("Quinlan").

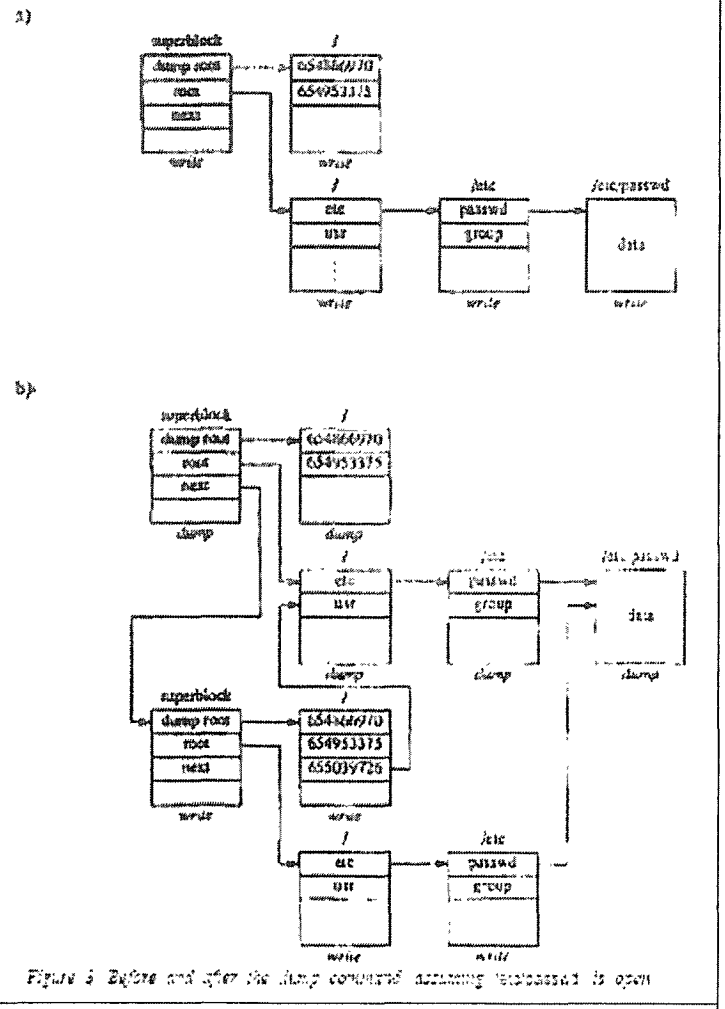
The pertinence and manner of applying Quinlan to claims 1-24 for which re-examination is requested is as follows:

Claims of '211 Patent	Quinlan
1. A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:	Quinlan teaches a file system stored in a magnetic disc cache memory and on a WORM optical disk storage system.
maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system; and	Quinlan teaches a file system with a Unix file hierarchy. Quinlan, p. 1291. A root inode with levels of indirection, pointing directly and indirectly to data blocks, is inherent in a Unix file hierarchy. Figure 3 on p. 1295 further teaches a root inode directly and indirectly pointing to blocks. The root block (root inode) "contains the directory entry for the root of the tree and all blocks can trace a path from this root." Quinlan, p. 1291.
maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Quinlan teaches the use of a cache maintaining in-core copies of a subset of the file system. Quinlan, p. 1290. Quinlan explains that a copy-on-write technique is used. Quinlan, pp. 1291-92. After a block is modified via copy-on-write in memory, its directory (inode) is also modified with a pointer to the new location. *Id.* For the system to operate, it is inherent that the root inode is cached in memory and modified as data is modified. Periodically, the Quinlan file system atomically advances from one consistent state to another by flushing the contents of the cache to disk, thereby forming a snapshot. Quinlan, p. 1292-95, Fig. 3. The active file system shares unmodified data blocks with the previous snapshots. Quinlan, p. 1292. "In effect, the snapshot and the file system will split apart as the file system is modified." Quinlan, p1294. After a snapshot is taken, the in-core root inode pointing to buffers and new blocks reflects changes from the previous consistent state: The root of the file system is moved to a write block. . . . Note that changes to the file system are only visible through the new root block; the file system remains unchanged when viewed through the old root." *Id.* This is illustrated in Fig. 3:

	
2. A method as in claim 1, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Quinlan teaches that the system advances from one consistent state to the next atomically, using the *dump* operation: "The dump is performed as an atomic operation." Quinlan, p. 1294.
3. A method as in claim 2, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Quinlan teaches that at the end of the *dump* operation the root of the file system (root inode) is updated so as to provide access to the snapshot. Quinlan, p. 1294. It is inherent that this operation occurs after all the other data is flushed to disk. This is also reflected in Fig. 3.
4. A method as in claim 3, wherein updating said on-disk root inode	Replicating the metadata, such as a root inode, is a technique that is well known in the art. In view of the

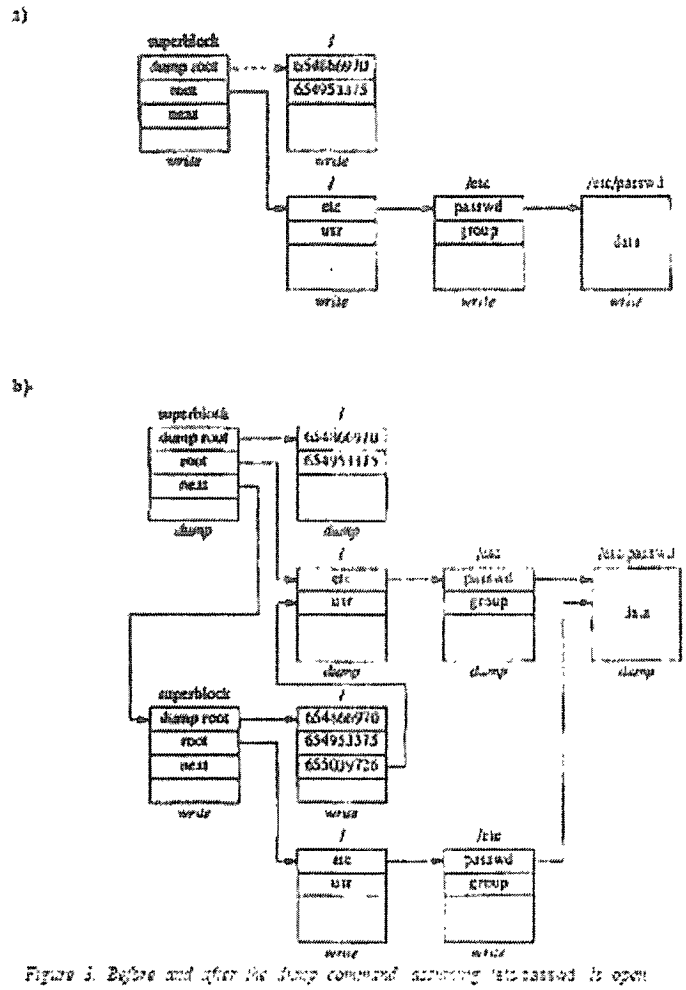
further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	knowledge in the art it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
5. A method as in claim 1, further comprising the step of creating a snapshot of said file system by copying only said on-disk root inode.	Quinlan teaches the creation of multiple snapshots. To do this, the root inode is updated so that it points to the most recent snapshot. Quinlan, p. 1294, Fig. 3.
6. A method as in claim 5, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Quinlan teaches that the efficiency of the file system is enhanced because "the number of blocks to write is minimal, as any block that has not changed since the previous dump will be shared in both snapshots and need not be written." Quinlan, p. 1295.
7. A method as in claim 1, further comprising the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Quinlan teaches the creation of multiple snapshots. To do this, the root inode is updated so that it points to the most recent snapshot. Quinlan, p. 1294, Fig. 3.
8. A method as in claim 7, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	The active file system shares unmodified data blocks with the previous snapshots. Quinlan, p. 1292. "In effect, the snapshot and the file system will split apart as the file system is modified." Quinlan, p1294. Quinlan further teaches that the efficiency of the file system is enhanced because "the number of blocks to write is minimal, as any block that has not changed since the previous dump will be shared in both snapshots and need not be written." Quinlan, p. 1295.
9. A device comprising: a processor; a memory; and	The file system taught in Quinlan is inherently executed on a processor with a memory.
a storage system including one or more hard disks;	Quinlan file system has data stored in cache memory and on a WORM disk storage system.
wherein said memory and said storage system store a file system; and	Quinlan file system has data stored in cache memory and on a WORM disk storage system.
wherein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including steps of (a)	Quinlan teaches a file system with a Unix file hierarchy. Quinlan, p. 1291. A root inode with levels of indirection, pointing directly and indirectly to data blocks, is inherent in a Unix file hierarchy. Figure 3 on p. 1295 further teaches a root inode directly and

maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	indirectly pointing to blocks. The root block (root inode) "contains the directory entry for the root of the tree and all blocks can trace a path from this root." Quinlan, p. 1291. Quinlan further teaches the use of a cache maintaining in-core copies of a subset of the file system. Quinlan, p. 1290. Quinlan explains that a copy-on-write technique is used. Quinlan, pp. 1291-92. After a block is modified via copy-on-write in memory, its directory (inode) is also modified with a pointer to the new location. *Id.* For the system to operate, it is inherent that the root inode is cached in memory and modified as data is modified. Periodically, the Quinlan file system atomically advances from one consistent state to another by flushing the contents of the cache to disk, thereby forming a snapshot. Quinlan, p. 1292-95, Fig. 3. The active file system shares unmodified data blocks with the previous snapshots. Quinlan, p. 1292. "In effect, the snapshot and the file system will split apart as the file system is modified." Quinlan, p1294. After a snapshot is taken, the in-core root inode pointing to buffers and new blocks reflects changes from the previous consistent state: The root of the file system is moved to a write block. . . . Note that changes to the file system are only visible through the new root block; the file system remains unchanged when viewed through the old root." *Id.* This is illustrated in Fig. 3:

	
10. A device as in claim 9, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Quinlan teaches that the system advances from one consistent state to the next atomically, using the *dump* operation: "The dump is performed as an atomic operation." Quinlan, p. 1294.
11. A device as in claim 10, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Quinlan teaches that at the end of the *dump* operation the root of the file system (root inode) is updated so as to provide access to the snapshot. Quinlan, p. 1294. It is inherent that this operation occurs after all the other data is flushed to disk. This is also reflected in Fig. 3.
12. A device as in claim 11, wherein updating said on-disk root inode	Replicating the metadata, such as a root inode, is a technique that is well known in the art. In view of the

further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	knowledge in the art it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
13. A device as in claim 9, wherein the instructions further comprise the step of creating a snapshot of said file system by copying only said on-disk root inode.	Quinlan teaches the creation of multiple snapshots. To do this, the root inode is updated so that it points to the most recent snapshot. Quinlan, p. 1294, Fig. 3.
14. A device as in claim 13, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Quinlan teaches that the efficiency of the file system is enhanced because "the number of blocks to write is minimal, as any block that has not changed since the previous dump will be shared in both snapshots and need not be written." Quinlan, p. 1295.
15. A device as in claim 9, wherein the instructions further comprise the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Quinlan teaches the creation of multiple snapshots. To do this, the root inode is updated so that it points to the most recent snapshot. Quinlan, p. 1294, Fig. 3.
16. A device as in claim 15, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	The active file system shares unmodified data blocks with the previous snapshots. Quinlan, p. 1292. "In effect, the snapshot and the file system will split apart as the file system is modified." Quinlan, p1294. Quinlan further teaches that the efficiency of the file system is enhanced because "the number of blocks to write is minimal, as any block that has not changed since the previous dump will be shared in both snapshots and need not be written." Quinlan, p. 1295.
17. An article of manufacture comprising a machine-readable storage medium storing instructions executable by a processor coupled to a memory and to a storage system, said storage system comprising one or more hard disks, said memory and said storage system storing a file system, wherein the instructions, when executed by the processor,	The Quinlan file system is inherently stored on machine readable medium in the form of instructions executable by a processor coupled to a memory and one or more hard disks. Quinlan teaches a file system with a Unix file hierarchy. Quinlan, p. 1291. A root inode with levels of indirection, pointing directly and indirectly to data blocks, is inherent in a Unix file hierarchy. Figure 3 on p. 1295 further teaches a root inode directly and

cause the processor to (a) maintain an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintain an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	indirectly pointing to blocks. The root block (root inode) "contains the directory entry for the root of the tree and all blocks can trace a path from this root." Quinlan, p. 1291. Quinlan further teaches the use of a cache maintaining in-core copies of a subset of the file system. Quinlan, p. 1290. Quinlan explains that a copy-on-write technique is used. Quinlan, pp. 1291-92. After a block is modified via copy-on-write in memory, its directory (inode) is also modified with a pointer to the new location. *Id.* For the system to operate, it is inherent that the root inode is cached in memory and modified as data is modified. Periodically, the Quinlan file system atomically advances from one consistent state to another by flushing the contents of the cache to disk, thereby forming a snapshot. Quinlan, p. 1292-95, Fig. 3. The active file system shares unmodified data blocks with the previous snapshots. Quinlan, p. 1292. "In effect, the snapshot and the file system will split apart as the file system is modified." Quinlan, p1294. After a snapshot is taken, the in-core root inode pointing to buffers and new blocks reflects changes from the previous consistent state: The root of the file system is moved to a write block. . . . Note that changes to the file system are only visible through the new root block; the file system remains unchanged when viewed through the old root." *Id.* This is illustrated in Fig. 3:

	
18. An article of manufacture as in claim 17, wherein the instructions further cause the processor to move atomically said file system on said storage system from said first consistent state to said second consistent state.	Quinlan teaches that the system advances from one consistent state to the next atomically, using the *dump* operation: "The dump is performed as an atomic operation." Quinlan, p. 1294.
19. An article of manufacture as in claim 18, wherein the instructions cause the processor to move atomically said file system on said storage system to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information	Quinlan teaches that at the end of the *dump* operation the root of the file system (root inode) is updated so as to provide access to the snapshot. Quinlan, p. 1294. It is inherent that this operation occurs after all the other data is flushed to disk. This is also reflected in Fig. 3.

from said incore root inode.	
20. An article of manufacture as in claim 19, wherein updating said on-disk root inode comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. In view of the knowledge in the art it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
21. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create a snapshot of said file system by copying only said on-disk root inode.	Quinlan teaches the creation of multiple snapshots. To do this, the root inode is updated so that it points to the most recent snapshot. Quinlan, p. 1294, Fig. 3.
22. An article of manufacture as in claim 21, wherein the instructions cause the processor to create said snapshot so that said snapshot and said file system share said first set of blocks on said storage system when said snapshot is created.	Quinlan teaches that the efficiency of the file system is enhanced because "the number of blocks to write is minimal, as any block that has not changed since the previous dump will be shared in both snapshots and need not be written." Quinlan, p. 1295.
23. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create plural snapshots of said file system by copying only said on-disk root inode at different times.	Quinlan teaches the creation of multiple snapshots. To do this, the root inode is updated so that it points to the most recent snapshot. Quinlan, p. 1294, Fig. 3.
24. An article of manufacture as in claim 23, wherein the instructions, when executed by the processor, cause the processor to create each one of said plural snapshots so that each one of said snapshots and said file system share said first set of blocks on said storage system when each one of said plural snapshots is created.	The active file system shares unmodified data blocks with the previous snapshots. Quinlan, p. 1292. "In effect, the snapshot and the file system will split apart as the file system is modified." Quinlan, p1294. Quinlan further teaches that the efficiency of the file system is enhanced because "the number of blocks to write is minimal, as any block that has not changed since the previous dump will be shared in both snapshots and need not be written." Quinlan, p. 1295.

Second Basis of Invalidity

The references applicable to the second basis of invalidity are:

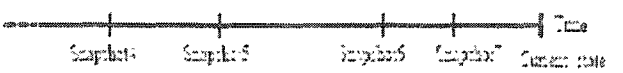
1. Popek, Walker, *The LOCUS Distributed System Architecture*, MIT Press, Cambridge, Mass., 1985 ("Popek").

2. Ylonen, *Concurrent Shadow Paging: A new Direction for Database Research*, Helsinki University of Technology, TKO-B86, 1992 ("Ylonen")

The pertinence and manner of applying Popek and Ylonen to claims 1-24 for which re-examination is requested is as follows:

Claims of '211 Patent	Popek and Ylonen
1. A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:	Popek teaches a LOCUS file system storing data in a memory and on hard disk. *See e.g.* Popek, Sec. 3.3, pp. 33-34, Sec. 3.4.6, pp. 46-48.
maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system; and	Popek teaches that LOCUS maintains an on-disk inode that contains "page numbers" or pointers to data blocks or to "intermediate node[s]" that point to data blocks Popek, Sec. 3.3, pp. 33-34, Sec. 3.4.6, pp. 46-48. "A file is composed of an inode and an associated ordered collection of data blocks." *Id.* at 34. "Additional mechanism is also present to support large files that are structured through indirect pages that contain page pointers." Popek, Sec. 3.4.6, p. 47. The logical pages or data blocks taught in Popek inherently represent a consistent state of the LOCUS file system.
maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Popek teaches that LOCUS maintains an "in-core copy of the disk inode." Popek, Sec. 3.4.6, p. 47. LOCUS uses a shadow paging mechanism (copy-on-write) to update files without overwriting data in place. *Id.* As logical pages (logical blocks) are updated, "it is necessary to keep track of where the old and the new pages are. The disk inode contains the old page numbers. The in-core copy of the disk inode starts with the old pages, but is updated with new page numbers as shadow pages are allocated." *Id.* As logical pages are updated, they are periodically flushed to a new location on disk. *Id.* As a result, the in-core inode points to a set of blocks on disk that includes the old unchanged blocks and newly flushed blocks containing

	changes, as well as to memory buffers that have not yet been flushed. The sum total of these buffers and blocks inherently represents a second consistent state of the LOCUS file system.
2. A method as in claim 1, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Popek teaches an "atomic commit operation" that moves the file system from one consistent state to the next. *See* Popek, Sec. 3.4.6, p. 47.
3. A method as in claim 2, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Popek teaches flushing changed logical pages to disk and then executing an atomic commit operation by "moving the in-core inode information to the disk inode." Popek, Sec. 3.4.6, p. 47.
4. A method as in claim 3, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. In view of the knowledge in the art, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
5. A method as in claim 1, further comprising the step of creating a snapshot of said file system by copying only said on-disk root inode.	Popek teaches that "[t]he important concept of atomically committing changes has been imported from the database world and integrated into LOCUS." Popek, Sec. 3.4.6, p. 46. Therefore, one of ordinary skill in the art would find motivation to look to the teachings of database art, such as Ylonen. Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be

	consistent as long as their pages are not freed." *Id.*
6. A method as in claim 5, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.
7. A method as in claim 1, further comprising the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
8. A method as in claim 7, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.
9. A device comprising: a processor; a memory; and	The file system taught in Popek is inherently executed on a processor with a memory.
a storage system including one or more	Popek teaches a LOCUS file system storing data in

hard disks;	a memory and on hard disk. *See e.g.* Popek, Sec. 3.3, pp. 33-34, Sec. 3.4.6, pp. 46-48.
wherein said memory and said storage system store a file system; and	The file system in Popek is stored in memory and on hard disk. *Id.*
wherein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including steps of (a) maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Popek teaches that LOCUS maintains an on-disk inode that contains "page numbers" or pointers to data blocks or to "intermediate node[s]" that point to data blocks Popek, Sec. 3.3, pp. 33-34, Sec. 3.4.6, pp. 46-48. "A file is composed of an inode and an associated ordered collection of data blocks." *Id.* at 34. "Additional mechanism is also present to support large files that are structured through indirect pages that contain page pointers." Popek, Sec. 3.4.6, p. 47. The logical pages or data blocks taught in Popek inherently represent a consistent state of the LOCUS file system. Popek further teaches that LOCUS maintains an "in-core copy of the disk inode." Popek, Sec. 3.4.6, p. 47. LOCUS uses a shadow paging mechanism (copy-on-write) to update files without overwriting data in place. *Id.* As logical pages (logical blocks) are updated, "it is necessary to keep track of where the old and the new pages are. The disk inode contains the old page numbers. The in-core copy of the disk inode starts with the old pages, but is updated with new page numbers as shadow pages are allocated." *Id.* As logical pages are updated, they are periodically flushed to a new location on disk. *Id.* As a result, the in-core inode points to a set of blocks on disk that includes the old unchanged blocks and newly flushed blocks containing changes, as well as to memory buffers that have not yet been flushed. The sum total of these buffers and blocks inherently represents a second consistent state of the LOCUS file system.
10. A device as in claim 9, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Popek teaches an "atomic commit operation" that moves the file system from one consistent state to the next. *See* Popek, Sec. 3.4.6, p. 47.
11. A device as in claim 10, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers	Popek teaches flushing changed logical pages to disk and then executing an atomic commit operation by "moving the in-core inode information to the disk inode." Popek, Sec. 3.4.6,

to said storage system before updating said on-disk root inode with information from said incore root inode.	p. 47.
12. A device as in claim 11, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. In view of the knowledge in the art, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.
13. A device as in claim 9, wherein the instructions further comprise the step of creating a snapshot of said file system by copying only said on-disk root inode.	Popek teaches that "[t]he important concept of atomically committing changes has been imported from the database world and integrated into LOCUS." Popek, Sec. 3.4.6, p. 46. Therefore, one of ordinary skill in the art would find motivation to look to the teachings of database art, such as Ylonen. Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
14. A device as in claim 13, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  FIGS. 2: Snapshots represent consistent database states as of some time in the past.

15. A device as in claim 9, wherein the instructions further comprise the step of creating plural snapshots of said file system by copying only said on-disk root inode at different times.	Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
16. A device as in claim 15, wherein when each one of said plural snapshots is created, each one of said snapshots and said file system share said first set of blocks on said storage system.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.
17. An article of manufacture comprising a machine-readable storage medium storing instructions executable by a processor coupled to a memory and to a storage system, said storage system comprising one or more hard disks, said memory and said storage system storing a file system, wherein the instructions, when executed by the processor, cause the processor to (a) maintain an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintain an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers	The file system of Popek is inherently stored on machine readable medium in the form of instructions executable by a processor coupled to a memory and one or more hard disks. Popek teaches that LOCUS maintains an on-disk inode that contains "page numbers" or pointers to data blocks or to "intermediate node[s]" that point to data blocks Popek, Sec. 3.3, pp. 33-34, Sec. 3.4.6, pp. 46-48. "A file is composed of an inode and an associated ordered collection of data blocks." *Id.* at 34. "Additional mechanism is also present to support large files that are structured through indirect pages that contain page pointers." Popek, Sec. 3.4.6, p. 47. The logical pages or data blocks taught in Popek inherently represent a consistent state of the LOCUS file system.

in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Popek further teaches that LOCUS maintains an "in-core copy of the disk inode." Popek, Sec. 3.4.6, p. 47. LOCUS uses a shadow paging mechanism (copy-on-write) to update files without overwriting data in place. *Id.* As logical pages (logical blocks) are updated, "it is necessary to keep track of where the old and the new pages are. The disk inode contains the old page numbers. The in-core copy of the disk inode starts with the old pages, but is updated with new page numbers as shadow pages are allocated." *Id.* As logical pages are updated, they are periodically flushed to a new location on disk. *Id.* As a result, the in-core inode points to a set of blocks on disk that includes the old unchanged blocks and newly flushed blocks containing changes, as well as to memory buffers that have not yet been flushed. The sum total of these buffers and blocks inherently represents a second consistent state of the LOCUS file system.
18. An article of manufacture as in claim 17, wherein the instructions further cause the processor to move atomically said file system on said storage system from said first consistent state to said second consistent state.	Popek teaches an "atomic commit operation" that moves the file system from one consistent state to the next. *See* Popek, Sec. 3.4.6, p. 47.
19. An article of manufacture as in claim 18, wherein the instructions cause the processor to move atomically said file system on said storage system to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Popek teaches flushing changed logical pages to disk and then executing an atomic commit operation by "moving the in-core inode information to the disk inode." Popek, Sec. 3.4.6, p. 47.
20. An article of manufacture as in claim 19, wherein updating said on-disk root inode comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Replicating the metadata, such as a root inode, is a technique that is well known in the art. In view of the knowledge in the art, it would have been obvious for one of ordinary skill to make a copy of the root inode to recover from crashes that corrupt the primary copy of the root inode.

21. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create a snapshot of said file system by copying only said on-disk root inode.	Popek teaches that "[t]he important concept of atomically committing changes has been imported from the database world and integrated into LOCUS." Popek, Sec. 3.4.6, p. 46. Therefore, one of ordinary skill in the art would find motivation to look to the teachings of database art, such as Ylonen. Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
22. An article of manufacture as in claim 21, wherein the instructions cause the processor to create said snapshot so that said snapshot and said file system share said first set of blocks on said storage system when said snapshot is created.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.
23. An article of manufacture as in claim 17, wherein the instructions further cause the processor to create plural snapshots of said file system by copying only said on-disk root inode at different times.	Ylonen teaches an atomic commit method for databases, wherein a page table pointer (equivalent to a root inode) is updated to atomically advance the database to the next consistent state. Ylonen, Sec. 2, pp. 1-2, Fig. 1. Ylonen further teaches taking one or more snapshots by saving the address of the page table and preventing freeing of data pages. Ylonen, Sec. 8, p. 4, Fig. 2. The page table address is in the page table pointer. Thus, every time the page table pointer is copied (an operation that occurs at every commit cycle), "[w]e can have

	an arbitrary number of snapshots at different times, and they will all be consistent as long as their pages are not freed." *Id.*
24. An article of manufacture as in claim 23, wherein the instructions, when executed by the processor, cause the processor to create each one of said plural snapshots so that each one of said snapshots and said file system share said first set of blocks on said storage system when each one of said plural snapshots is created.	Ylonen teaches a shadow paging (copy-on-write) technique that "never modifies accessible pages," therefore a snapshot and the active file system share unmodified blocks. Ylonen, Sec. 8, p. 4, Figs. 1 and 2.  Figure 2: Snapshots represent consistent database states as of some time in the past.

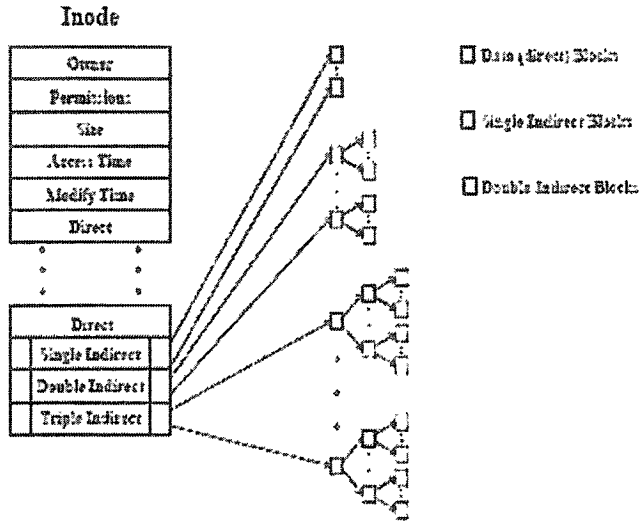
Third Basis of Invalidity

The reference applicable to the third basis of invalidity is:

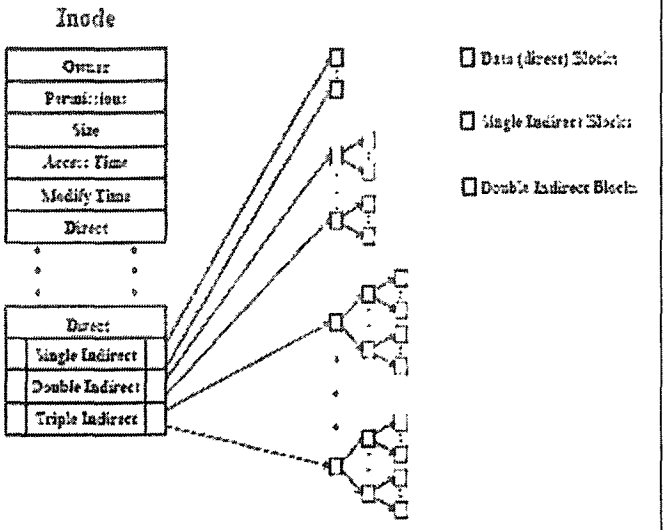
1. Margo Ilene Seltzer , *File System Performance and Transaction Support*, Doctoral Dissertation, UC Berkeley, 1992 ("Seltzer").

The pertinence and manner of applying Seltzer to claims 1-6, 9-14, 17-22 for which re-examination is requested is as follows:

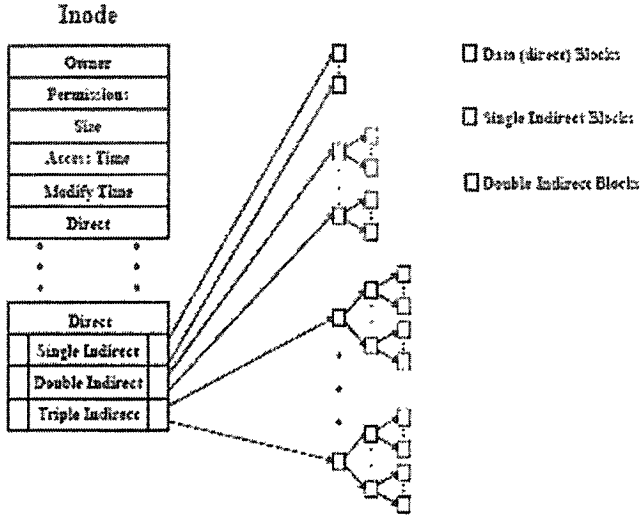
Claims of '211 Patent	Seltzer
1. A method of maintaining a file system stored in a memory and on a storage system that includes one or more hard disks, said method comprising steps of:	Seltzer presents an LFS file system, with data stored in memory and on a disk storage system. *See e.g.* Seltzer, p. 59.
maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system; and	Seltzer teaches that the LFS file system maintains an on-disk root inode that points directly and indirectly to a first set of storage blocks on disk in a consistent state. *See e.g.* Sec. 5.2.1.3 at p. 59, Sec. 6.1.1 at pp.70-71. This is further illustrated in Fig. 5-5:

	 Figure 5-5: File Index Structure (inode).
maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Seltzer teaches an in-memory inode containing the physical representation of the file system layout. Seltzer, Sec. 5.2.1.3, p. 59, Fig. 5-5. The in-memory inode "additionally includes lists of buffers and links to other inodes. . . . The in-memory inode is extended to have a list of transaction-protected buffers in addition to its clean and dirty buffer lists." *Id.* "When a file is written, the new data blocks are appended to the log, and the index structure and indirect blocks are modified (in memory) to contain the new disk address of the newly written block." Seltzer, Sec. 4.1, pp. 31-32. Therefore, Seltzer teaches that the in-memory inode represents a consistent up-to-date state of the file system, pointing directly and indirectly to buffers containing new or modified data as well to the unchanged blocks of data. Seltzer further teaches that "[w]hen writing, LFS gathers many dirty pages and prepares to write them to disk sequentially in the next available segment. At that time, LFS sorts the blocks by logical block number, assigns them disk addresses, and updates the meta-data to reflect their addresses." Seltzer, Sec. 6.1.1, p. 71. Once some buffers holding dirty pages (changed data) are flushed to disk blocks, they, together with unchanged blocks extant on disk, constitute a second set of blocks. This second set of blocks, in combination with un-flushed dirty buffers, represent the up-to-date file system in a consistent state.

2. A method as in claim 1, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.*
3. A method as in claim 2, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Seltzer teaches that the new *ifile* inode is flushed to disk after the dirty buffers are flushed. *See* Seltzer, Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88.
4. A method as in claim 3, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Seltzer teaches that the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Seltzer, Sec. 6.1.1, p. 71. In view of this teaching, it would have been obvious for one of ordinary skill in the art to replicate the root inode to recover from crashes that corrupt the primary copy of the root inode.
5. A method as in claim 1, further comprising the step of creating a snapshot of said file system by copying only said on-disk root inode.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.*
6. A method as in claim 5, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.* The checkpoint inherently shares unmodified blocks with the active file system.
9. A device comprising: a processor; a memory; and	The file system taught in Seltzer is inherently executed on a processor with a memory.
a storage system including one or more hard disks;	Seltzer presents an LFS file system, with data stored in memory and on a disk storage system. *See e.g.* Seltzer, p. 59.
wherein said memory and said storage system store a file system; and	Seltzer presents an LFS file system, with data stored in memory and on a disk storage system. *See e.g.* Seltzer, p. 59.

wherein said memory also stores information including instructions executable by said processor to maintain said file system, the instructions including steps of (a) maintaining an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintaining an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	Seltzer teaches that the LFS file system maintains an on-disk root inode that points directly and indirectly to a first set of storage blocks on disk in a consistent state. *See e.g.* Sec. 5.2.1.3 at p. 59, Sec. 6.1.1 at pp.70-71. This is further illustrated in Fig. 5-5:  **Figure 5-5: File Index Structure (inode).** Seltzer teaches an in-memory inode containing the physical representation of the file system layout. Seltzer, Sec. 5.2.1.3, p. 59, Fig. 5-5. The in-memory inode "additionally includes lists of buffers and links to other inodes. . . . The in-memory inode is extended to have a list of transaction-protected buffers in addition to its clean and dirty buffer lists." *Id.* "When a file is written, the new data blocks are appended to the log, and the index structure and indirect blocks are modified (in memory) to contain the new disk address of the newly written block." Seltzer, Sec. 4.1, pp. 31-32. Therefore, Seltzer teaches that the in-memory inode represents a consistent up-to-date state of the file system, pointing directly and indirectly to buffers containing new or modified data as well to the unchanged blocks of data. Seltzer further teaches that "[w]hen writing, LFS gathers many dirty pages and prepares to write them to disk sequentially in the next available segment. At that time, LFS sorts the blocks by logical block number, assigns them disk addresses, and updates the meta-data to reflect their addresses." Seltzer, Sec. 6.1.1, p. 71. Once

	some buffers holding dirty pages (changed data) are flushed to disk blocks, they, together with unchanged blocks extant on disk, constitute a second set of blocks. This second set of blocks, in combination with un-flushed dirty buffers, represent the up-to-date file system in a consistent state.
10. A device as in claim 9, wherein said file system on said storage system always moves atomically from said first consistent state to said second consistent state.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.*
11. A device as in claim 10, wherein said file system on said storage system moves atomically to said second consistent state by flushing said changes from said buffers to said storage system before updating said on-disk root inode with information from said incore root inode.	Seltzer teaches that the new *ifile* inode is flushed to disk after the dirty buffers are flushed. *See* Seltzer, Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88.
12. A device as in claim 11, wherein updating said on-disk root inode further comprises updating said on-disk root inode and then a copy of said on-disk root inode such that if updating said on-disk root inode is interrupted, said copy of said on-disk root inode still points to said first consistent state of said file system.	Seltzer teaches that the superblock is replicated to allow recovery from crashes that corrupt the primary copy of the superblock. Seltzer, Sec. 6.1.1, p. 71. In view of this teaching, it would have been obvious for one of ordinary skill in the art to replicate the root inode to recover from crashes that corrupt the primary copy of the root inode.
13. A device as in claim 9, wherein the instructions further comprise the step of creating a snapshot of said file system by copying only said on-disk root inode.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.*
14. A device as in claim 13, wherein when said snapshot is created, said snapshot and said file system share said first set of blocks on said storage system.	Seltzer teaches that the LFS file system atomically advances when the *ifile* inode is committed to disk. *See e.g.* Seltzer Sec. 6.1.1, pp. 71-72, Sec. 6.3.3, p. 88. A consistent checkpoint is created when the *ifile* inode is flushed to disk. *Id.* The checkpoint inherently shares unmodified blocks with the active file system.
17. An article of manufacture comprising a	The file system of Seltzer is inherently stored on

machine-readable storage medium storing instructions executable by a processor coupled to a memory and to a storage system, said storage system comprising one or more hard disks, said memory and said storage system storing a file system, wherein the instructions, when executed by the processor, cause the processor to (a) maintain an on-disk root inode on said storage system, said on-disk root inode pointing directly and indirectly to a first set of blocks on said storage system that store a first consistent state of said file system, and (b) maintain an incore root inode in said memory, said incore root inode pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system, said buffers and said second set of blocks storing data and metadata for a second consistent state of said file system, said second set of blocks including at least some blocks in said first set of blocks, with changes between said first consistent state and said second consistent state being stored in said buffers and in ones of said second set of blocks not pointed to by said on-disk inode.	machine readable medium in the form of instructions executable by a processor coupled to a memory and one or more hard disks. Seltzer teaches that the LFS file system maintains an on-disk root inode that points directly and indirectly to a first set of storage blocks on disk in a consistent state. *See e.g.* Sec. 5.2.1.3 at p. 59, Sec. 6.1.1 at pp.70-71. This is further illustrated in Fig. 5-5:  **Figure 5-5: File Index Structure (inode).** Seltzer teaches an in-memory inode containing the physical representation of the file system layout. Seltzer, Sec. 5.2.1.3, p. 59, Fig. 5-5. The in-memory inode "additionally includes lists of buffers and links to other inodes. . . . The in-memory inode is extended to have a list of transaction-protected buffers in addition to its clean and dirty buffer lists." *Id.* "When a file is written, the new data blocks are appended to the log, and the index structure and indirect blocks are modified (in memory) to contain the new disk address of the newly written block." Seltzer, Sec. 4.1, pp. 31-32. Therefore, Seltzer teaches that the in-memory inode represents a consistent up-to-date state of the file system, pointing directly and indirectly to buffers containing new or modified data as well to the unchanged blocks of data. Seltzer further teaches that "[w]hen writing, LFS gathers many dirty pages and prepares to write them to disk sequentially in